Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №6

Выполнила:

Студент 3 курса

Группы ПО-5

Нерода А.А.

Проверил:

Крощенко А.А.

Брест 2021

# Вариант-10

**Цель работы:** приобрести навыки применения паттернов проектирования при решении практических задач с использованием языка Java.

**Задание 1.** Заводы по производству автомобилей. Реализовать возможность создавать автомобили различных типов на различных заводах..

**Код программы**

public class Program {

    public static void main(String[] args) {

            Car electricCar = new Car(new ElectricCarFactory());

            electricCar.Drive();

            electricCar.Fuel();

            Car gasolineCar = new Car(new GasolineCarFactory());

            gasolineCar.Drive();

            gasolineCar.Fuel();

    }

}

    abstract class Fuel

    {

        public abstract void ToFill();

    }

    abstract class Movement

    {

        public abstract void Move();

    }

    class NuclearFuel extends Fuel

    {

        @Override

        public void ToFill()

        {

            System.out.println("Заправляем ядерным топливом!");

        }

    }

    class Gasoline extends Fuel

    {

        @Override

        public void ToFill()

        {

            System.out.println("Заправляем бензином!");

        }

    }

    class Electric extends Fuel

    {

        @Override

        public void ToFill()

        {

            System.out.println("Поставили на зарядку!");

        }

    }

    class FlyMovement extends Movement

    {

        @Override

        public void Move()

        {

            System.out.println("Летим");

        }

    }

    class DriveMovement extends Movement

    {

        @Override

        public void Move()

        {

            System.out.println("Едем");

        }

    }

    class NuclearFuelCarFactory extends CarFactory

    {

        @Override

        public Movement CreateVehicle()

        {

            return new DriveMovement();

        }

        @Override

        public Fuel FuelVehicle()

        {

            return new NuclearFuel();

        }

    }

    class GasolineCarFactory extends CarFactory

    {

        @Override

        public Movement CreateVehicle()

        {

            return new DriveMovement();

        }

        @Override

        public Fuel FuelVehicle()

        {

            return new Gasoline();

        }

    }

    class ElectricCarFactory extends CarFactory

    {

        @Override

        public Movement CreateVehicle()

        {

            return new FlyMovement();

        }

        @Override

        public Fuel FuelVehicle()

        {

            return new Electric();

        }

    }

    abstract class CarFactory

    {

        public abstract Movement CreateVehicle();

        public abstract Fuel FuelVehicle();

    }

    class Car

    {

        private Fuel fuel;

        private Movement car;

        public Car(CarFactory factory)

        {

            car = factory.CreateVehicle();

            fuel = factory.FuelVehicle();

        }

        public void Drive()

        {

            car.Move();

        }

        public void Fuel()

        {

            fuel.ToFill();

        }

    }

**Результаты работы**

Летим

Поставили на зарядку!

Едем

Заправляем бензином!

**Задание 2. У**четная запись покупателя книжного интернет-магазина. Предусмотреть различные уровни учетки в зависимости от активности покупателя. Дополнительные уровни добавляют функциональные возможности и открывают доступ к уникальным предложениям.

**Код программы**

import java.util.ArrayList;

public class AccRun {

    public static void main(String[] args)

    {

        ArrayList<Human> People = new ArrayList<Human>();

        Human Ivan = new Low();

        Human Petr = new Average();

        Human Vera = new High();

        People.add(Ivan);

        People.add(Petr);

        People.add(Vera);

        System.out.println("Акция на новый год и Рождество");

        for (Human buyer: People)

        {

            System.out.print("Читатель " + buyer.getName() + " ");

            if(buyer.GetActivity() > 9)

            {

                buyer = new Discount(buyer);

                buyer = new Gift(buyer);

                System.out.println("получает на " + buyer.getName());

            }

            else if (buyer.GetActivity() < 5) { System.out.println("ничего не получает"); }

            else

            {

                buyer = new Discount(buyer);

                System.out.println("получает на " + buyer.getName());

            }

        }

    }

}

abstract class Human

{

    public String name;

    public Human(String n)

    {

        this.name = n;

    }

    public String getName() { return this.name; }

    public abstract int GetActivity();

}

class High extends Human

{

    public High()

    {

        super("высокий уровень");

    }

    @Override

    public int GetActivity()

    {

        return 10;

    }

}

class Average extends Human

{

    public Average()

    {

        super("средний уровень");

    }

    @Override

    public int GetActivity()

    {

        return 8;

    }

}

class Low extends Human

{

    public Low()

    {

        super("низкий уровень");

    }

    @Override

    public int GetActivity()

    {

        return 4;

    }

}

abstract class HumanDecorator extends Human

{

    protected Human human;

    public HumanDecorator(String name, Human human)

    {

        super(name);

        this.human = human;

    }

}

class Discount extends HumanDecorator

{

    public Discount(Human p)

    {

        super(p.getName() + " скидку 20% ", p);

    }

    @Override

    public int GetActivity()

    {

        return human.GetActivity() - 3;

    }

}

class Gift extends HumanDecorator

{

    public Gift(Human p)

    {

        super(p.getName() + "и подарок", p);

    }

    @Override

    public int GetActivity()

    {

        return human.GetActivity() - 5;

    }

}

**Результаты работы**

Акция на новый год и Рождество

Читатель низкий уровень ничего не получает

Читатель средний уровень получает на средний уровень скидку 20%

Читатель высокий уровень получает на высокий уровень скидку 20% и подарок

**Задание 3.** Проект «Банкомат». Предусмотреть выполнение основных операций (ввод пин-кода, снятие суммы, завершение работы) и наличие различных режимов работы (ожидание, аутентификация, выполнение операции, блокировка – если нет денег). Атрибуты: общая сумма денег в банкомате, ID.

**Код программы**

import java.util.Scanner;

public class ATM {

    public static void main(String[] args) {

        Scanner in = new Scanner(System.in);

        System.out.print("Введите сумму денег в банкомате: ");

        int sum = in.nextInt();

        in.nextLine();

        Bankomat bankomat = new Bankomat(new Waiting(), sum);

        bankomat.Info();

        bankomat.EnterPin();

        bankomat.WithdrawalAmount();

        bankomat.CloseDown();

        in.close();

    }

}

class Bankomat

{

    public IATMState State;

    public boolean ID;

    public int sumMoney;

    public IATMState getState() { return State; }

    public void setState(IATMState state) { State = state; }

    public Bankomat(IATMState state, int summoney)

    {

        State = state;

        sumMoney = summoney;

    }

    public void EnterPin()

    {

        State.EnterPIN(this);

    }

    public void WithdrawalAmount()

    {

        State.WithdrawalAmount(this);

    }

    public void CloseDown()

    {

        State.CloseDown(this);

    }

    public void Info()

    {

        System.out.println("ID=" + ID + " и SUM=" + sumMoney);

    }

}

interface IATMState

{

    void EnterPIN(Bankomat bankomat);

    void WithdrawalAmount(Bankomat bankomat);

    void CloseDown(Bankomat bankomat);

}

class Waiting implements IATMState

{

    @Override

    public void EnterPIN(Bankomat bankomat)

    {

        if (bankomat.sumMoney == 0)

        {

            System.out.println("Блокировка...");

            bankomat.State = new Lock();

        }

        else

        {

            System.out.println("Аутентификация...");

            bankomat.State = new Authentication();

            bankomat.ID = true;

        }

    }

    @Override

    public void WithdrawalAmount(Bankomat bankomat)

    {

        System.out.println("Для выполенения операции выполните вход");

    }

    @Override

    public void CloseDown(Bankomat bankomat)

    {

        System.out.println("Завершение работы...");

        bankomat.ID = false;

    }

}

class Authentication implements IATMState

{

    @Override

    public void EnterPIN(Bankomat bankomat)

    {

        System.out.println("Вход уже выполнен");

    }

    @Override

    public void WithdrawalAmount(Bankomat bankomat)

    {

        if (bankomat.sumMoney == 0)

        {

            System.out.println("Блокировка...");

            bankomat.State = new Lock();

        }

        else

        {

            System.out.println("Сумма снята...");

            bankomat.State = new ExecuteOperation();

        }

    }

    @Override

    public void CloseDown(Bankomat bankomat)

    {

        System.out.println("Завершение работы...");

        bankomat.State = new Waiting();

        bankomat.ID = false;

    }

}

class ExecuteOperation implements IATMState

{

    @Override

    public void EnterPIN(Bankomat bankomat)

    {

        System.out.println("Аутентификация...");

        bankomat.State = new Authentication();

    }

    @Override

    public void WithdrawalAmount(Bankomat bankomat)

    {

        System.out.println("Введите пин для продолжения");

    }

    @Override

    public void CloseDown(Bankomat bankomat)

    {

        System.out.println("Завершение работы...");

        bankomat.State = new Waiting();

        bankomat.ID = false;

    }

}

class Lock implements IATMState

{

    @Override

    public void EnterPIN(Bankomat bankomat)

    {

        System.out.println("Банкомат заблокирован! Нет денег");

    }

    @Override

    public void WithdrawalAmount(Bankomat bankomat)

    {

        System.out.println("Банкомат заблокирован! Нет денег");

    }

    @Override

    public void CloseDown(Bankomat bankomat)

    {

        System.out.println("Завершение работы...");

        System.exit(0);

    }

}

**Результаты работы**

Введите сумму денег в банкомате: 0

ID=false и SUM=0

Блокировка...

Банкомат заблокирован! Нет денег

Завершение работы...

Введите сумму денег в банкомате: 2

ID=false и SUM=2

Аутентификация...

Сумма снята...

Завершение работы...

**Вывод:** в ходе работы приобрела навыки применения паттернов проектирования при решении практических задач с использованием языка Java.